

Før vi rigtig starter

Gå til <https://editor.p5js.org> og opret en konto, ved at trykke på knappen: **Sign up**

Første program

Tast følgende eksempel ind webeditoren:

```

kode der står under setup()      → function setup() {
bliver kørt én gang           }           ← forbereder et virtuelt "lærred" med en bredde på
                                            640 pixels og højde på 480 pixels.

kode der står under draw()       → function draw() {
bliver kørt igen og igen        }           ← Giver baggrunden en farve (grå)
                                            background(220);
                                            circle(100,200,150);
                                            line(80,250,120,250);
                                            ellipse(80,190,10,20);
                                            ellipse(120,190,10,20);
                                            triangle(100,200,90,230,110,230);
}

```

Klik på  , og se hvad der sker.

Gem projektet og kald det "tegning".

Tilføj nu følgende til draw-funktionen:

```

fjerner omrids på figur      → noStroke();
sætter omrids til en farve (sort) → stroke(0);
                                         triangle(400,250,500,100,600,250);
                                         triangle(400,150,600,150,500,300);

```

Og:

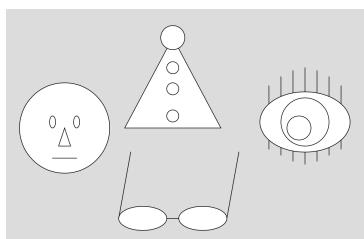
```

stroke(220);
ellipse(300, 195, 90,110);
ellipse(300,245,40,10);
ellipse(300,250,40,10);
ellipse(300,255,40,10);
ellipse(300,260,40,10);
ellipse(300,265,40,10);
ellipse(300,270,25,10);
stroke(0);

```

Opgave

Tegn 2-3 simple figurer - find gerne på dine egne!



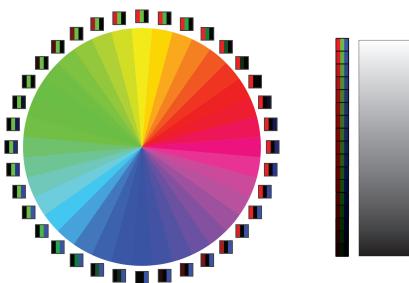
Farver

Kodeeksemplet er med en enkelt cirkel, men du skal bare lade de figurer du har tegnet stå.
 Skriv følgende ind i din kode:

```
function setup() {
  createCanvas(640, 480);
  colorYellow = color(255, 255, 0); ← gemmer en farve med navnet "colorYellow"
}

function draw() {
  background(220);
  fill(colorYellow); ← fylder figurene med en farve (colorYellow)
  circle(100,200,150);
}
```

For at angive en farve skal farvens RGB-værdier bestemmes. RGB står for red-green-blue.
 Man skriver **color(red , green ,blue)** hvor red, green og blue er et tal mellem 0 og 255.



Med **fill()** styrer hvilken farve du fylder dine figurer ud med. Du kan lave farver du kan genbruge i **setup()**, eller du kan skrive (r,g,b)-værdierne direkte som talværdier.

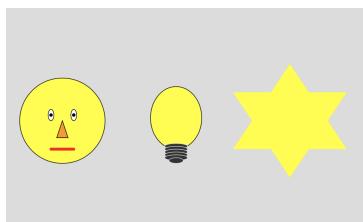
Du kan også skrive et enkelt tal for gråskala.

```
function draw() {
  background(220);
  fill(colorYellow); ← fylder figurene med en farve (colorYellow)
  circle(100,200,150);
  stroke(255,0,0);
  line(80,250,120,250);
  stroke(0);
  ellipse(80,190,10,20);
  ellipse(120,190,10,20);
  fill(255,150,0); ← fylder figurene med en farve (orange)
  triangle(100,200,90,230,110,230);
}
```

sætter omrids til en farve (rød)
 sætter omrids til en farve (sort)

Opgave

Giv din tegning noget farve.



På <https://p5js.org/reference/> kan du finde en oversigt over de vigtigste funktioner i p5js og en forklaring med eksempler på hvordan de bruges.

Funktioner

Åben p5js i et nyt faneblad og start en ny sketch.

Gem den nye sketch og kald den "funktioner".

Skriv følgende i din nye sketch:

```
function setup() {
  createCanvas(640, 480);
  colorYellow = color(255,255,0);
}

function draw() {
  background(220);
  star();
}

function star(){
  noStroke();
  fill(colorYellow);
  triangle(400,250,500,100,600,250);
  triangle(400,150,600,150,500,300);
}
```

For at kunne bevare overblikket i **draw()**, samles noget af koden i en funktion.

En funktion laves ved at man skriver:



function minfunktion() { kode }

Imellem de to krøllede parenteser {} skrives al den kode der skal køres når man kalder funktionen.

Nu kan man nemt og overskueligt få tegnet sin tegning ved at skrive funktionens navn, efterfulgt af (); under funktionen **draw()**

Opgave

Skriv en funktion til hver af dine tegninger og få dem tegnet kun ved at skrive funktionernes navne under **draw()**

```
function draw() {
  background(220);
  star();
  face();
  hat();
  briller();
  lightbulb();
  eye();
}
```

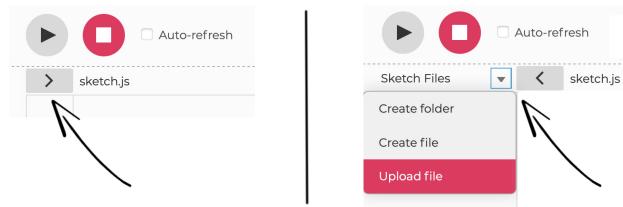
Billeder

Download dette billede af fyrværkeri - det ligger på wikicommons, som er et godt sted at lede efter billeder hvis man ikke vil overtræde nogens copyright:



https://upload.wikimedia.org/wikipedia/commons/0/04/Firework_transparent.png

Åbn sketchfiles, vælg "Upload file" og find fyrværkeribilledet der hvor du gemte det.



Skriv følgende i din kode:

```
function setup() {
  createCanvas(640, 480);
  firework = loadImage("firework_transparent.png"); ← indlæser billedet og giver det et navn
  colorYellow = color(255,255,0);
}

function draw() {
  background(220);
  star();
  image(firework,0,0,640,480); ← tegner billedet
}
```

Når du tegner billedet under **draw()** bruger du en funktion der hedder **image()**.

I parenteserne efter **image** står først navnet på dit billede, de to tal der står efter navnet, er x og y koordinaten til øverste venstre hjørne. Du behøver ikke at give flere tal med, men hvis dit billede er for stort, kan du give to tal mere med, nemlig den ønskede bredde og højde.

Opgave

Find selv et billede og sæt det ind i din kode.

Video

Vi bruger den indbyggede funktion **createCapture(VIDEO)** til at starte vores webcam. Skriv følgende kode ind:

```
function setup() {  
  createCanvas(640, 480);  
  firework = loadImage("Firework_transparent.png");  
  colorYellow = color(255,255,0);  
  video = createCapture(VIDEO);  
  video.size(640, 480);  
  video.hide();  
}  
  
function draw() {  
  background(220);  
  image(video,0,0);  
  star();  
  image(firework,0,0,640,480);  
}
```

Størrelsen på videooptagelsen sættes til samme størrelse som vores lærred. Sidste linje i **setup()** skjuler et preview af videoen, men den optager stadig.

For at vise videoen på lærredet, skal den tegnes i **draw()**. Det foregår på samme måde som når man tegner et billede. De to tal efter navnet er koordinaterne til øverste venstre hjørne. Vi har allerede angivet størrelsen, så det behøver vi ikke at gøre igen.

Opgave

Sæt de forskellige elementer vi har arbejdet med indtil nu sammen, så der både bliver tegnet video, billeder og de tegninger du har lavet med kode. Fordel tegninger og billeder lidt dekorativt omkring dit videobillede.



Start virtuelt kamera

p5.js:

Start med at gemme din sketch, hvis du ikke allerede har gjort det. Dernæst under **File** vælges **Share** og **Fullscreen**. Åbn linket i et nyt vindue.

OBS:

Download OBS - Open Broadcaster Software her: <https://obsproject.com>
Installer og åbn.

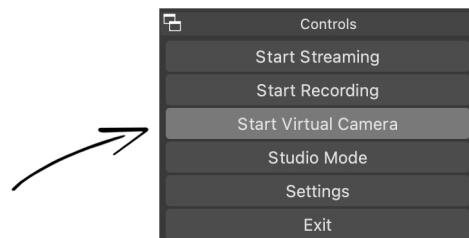
Tryk på + for at tilføje en ny **source/kilde** og vælg **Window capture/vindue optagelse**.

Vælg det vindue der kører din kode.



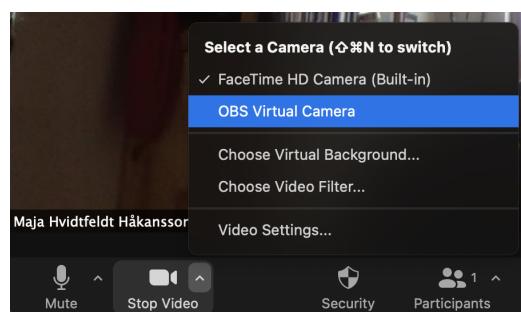
Beskær billedet i OBS ved at holde **Option-tasten** 0 inde på mMc eller Alt inde på Windows og træk i de røde punkter i rammen. Klip til så det kun er videobilledet der ses. Du skalerer ved at trække i de røde punkter uden at Option/Alt er holdt nede.

Nu mangler du kun at starte virtuelt kamera i OBS (under **Controls/Styring**)



Zoom:

Dit virtuelle kamera skulle gerne dukke op som et kamera der kan vælges i Zoom. Sandsynligvis skal Zoom genstartes.



Teachable Machine

Der kan være lidt forskel på hvordan Teachable Machine virker i de forskellige browsere. Hvis det ikke fungerer optimalt, så prøv at bruge en anden browser (Vi har testet at det virker i Chrome og Firefox).

Gå til <https://teachablemachine.withgoogle.com>

Tryk på den store blå knap for at komme i gang: 

Klik på **Image Project** for at lave en model der er baseret på billeder:



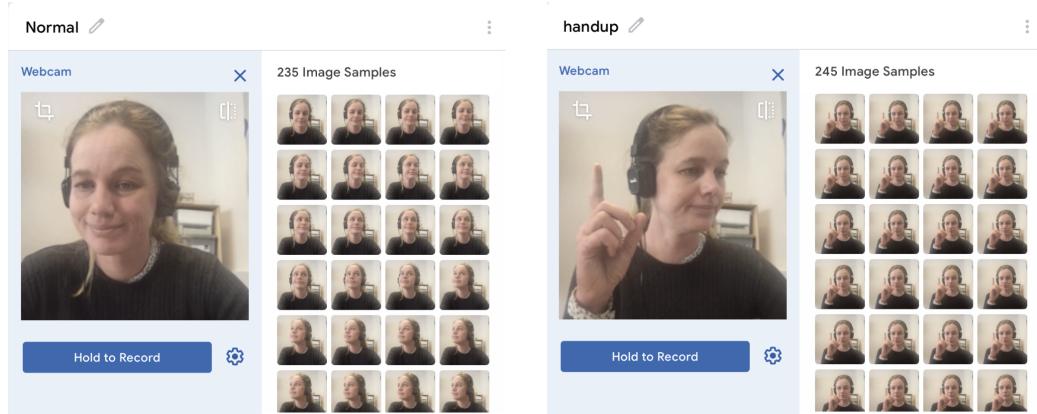
Image Project

Teach based on images, from files or your webcam.

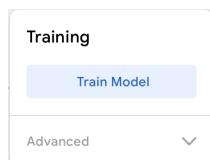
Start med at omdøbe **Class 1** til **Normal** og klik derefter på: 

Hold knappen  nede mens du tager en masse billeder hvor du ser "normal" ud. Drej dig lidt, kig lidt til siden og læn dig frem, som du ville gøre når du normalt deltager i virtuel undervisning.

Omdøb nu **Class 2** til **handup** og tag en serie billeder hvor du har hånden oppe. Du skal tage omkring 200 billeder i hver klasse.



Derefter klikker du på **Train model** og venter tålmodigt på at den bliver færdig.



Når modellen er trænet, kan du se hvor godt det virker med de billeder du har givet.



Den skulle gerne kunne skelne med en 95-100% sikkerhed - ellers må du tage nye billeder og sikre dig der er forskel på de billeder der ligger i de to klasser.

Exporter model

For at kunne bruge din model skal du klikke på **Export Model** Preview Export Model og derefter på **Upload my model** Upload (shareable link) Download Upload my model

Du får nu et link til din model. Kopier linket.

Tilhørende kode i p5.js

I et nyt faneblad åbner du dette link:

<https://editor.p5js.org/MajaHvidtfeldt/sketches/QZGHqB-fW>

Sketchen indeholder den kode og de funktioner der skal til for at vores billedegenkendelse virker.

Sæt dit link fra Teachable Machine ind i koden

```
imageModelURL = 'dit link copypastes ind her';
```

Hvis du bruger Windows, skal du være opmærksom på at at du ikke kan have Teachable Machine kørende i baggrunden samtidig med at du bruger video i p5.js

Opgave

Du skal flytte alle de funktioner du har lavet over i den nye sketch. Husk også at flytte det du har skrevet under **setup**. Du skal også uploade eventuelle billedfiler igen.

Du skal *ikke* flytte det du eventuelt har stående under **draw()**.



Tjek betingelse med *if-statement*

Når du kører din kode nu, vil classificeren automatisk løbende lægge navnet på det den genkender dit billede som, ind i **label**, så hvis vi tjekker hvad label er, kan vi kæde det sammen med bestemte handlinger.



Vi havde som udgangspunkt 2 klasser **Normal** og **handup**. Jeg kunne godt tænke mig at det ansigt jeg tidligere har tegnet bliver vist, når billedet klassificeres som "Normal", og at der kommer store røde spørgsmålstegn op, når jeg rækker fingeren i vejret.

```
function draw() {
  background(0);
  image(video, 0, 0);

  if (label == "Normal"){
    face();
  }

  if (label == "handup"){
    fill(255,0,0);
    textSize(120);
    text("? ? ? ? ? ?", 40,100);
  }
}
```

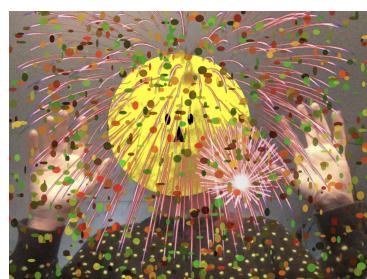
Nu skal du bare, på samme måde som tidligere starte det virtuelle kamera i OBS - og dine videomøder vil aldrig blive det samme igen!!

Opgave

Du kan træne en ny model med flere klasser, eksportere den og erstatte **imageModelURL**-linket i din kode med linket til den nye model.

Du skal bare lave et **if-statement** til hver af dine klasser.

Måske er wauw-effekten størst hvis der ikke sker noget når **label == "Normal"**, men du bestemmer helt selv. Fra nu af og resten af workshoppen er der kun een opgave - at have det sjovt med billeder og kode!



APPENDIX

Her er samlet nogle af de ting vi overvejede at have med i workshoppen, men var nødt til at pille ud. Det er ikke gennemarbejdede ark, men vi tænkte at de er jer der er blevet nysgerrige på mere, måske kunne komme skridte videre ved hjælp af dem.

Funktion med input

Når man laver funktioner kan man give dem input.

```
17 | function gulcirkel(x,y){  
18 |   fill(colYellow);  
19 |   stroke(0);  
20 |   strokeWeight(4);  
21 |   circle(x,y,200);  
22 | }  
23 |  
24 | function draw() {  
25 |   background(0);  
26 |   gulcirkel(110,30);  
27 |   gulcirkel(240,130);  
28 |   gulcirkel(200,200);  
29 |   gulcirkel(300,300);  
30 | }
```



Med overstående funktion kan jeg nemt tegne gule cirkle, lige hvor jeg

Du kan give din funktion ligeså mange input du har lyst til - og du må også gerne bruge navne i stedet for x, y, z. F.eks kunne man godt forestille sig at funktionen `gulcirkel` også tog et input der hed "diameter" og "farve": `gulcirkel(x, y, diameter, farve)`

Mikrofon

Under setup tilføjes:

```
18 |   mic = new p5.AudioIn();  
19 |   mic.start();
```

Nu lytter din computer med når du køre koden - hvis du altså husker at give den tilladelse. For f.eks. at få vist lydstyrken kan du nu i `draw()` skrive:

```
function draw() {  
  background(255);  
  let vol = mic.getLevel();  
  text(str(vol), 10, 30)  
}
```

En funktion der tegner med lyd

Det er fristende at bruge mikrofonen til at generere noget på skærmen. Her er et simpelt eksempel på en funktion man kunne lave:

```
function strip(){
    let volume = mic.getLevel();
    strokeWeight(10);
    r = int(volume*3000);
    g = int(255-(volume*3000));
    stroke(r,g,0);
    //placer en linje midt i billedet
    line(width/2, height, width/2, height-(volume * 5000));
    //placer en linje lidt til højre for midten
    line(width/2+40, height, width/2+40, height-(volume * 5000));
    //placer en linje lidt til venstre for midten
    line(width/2-40, height, width/2-40, height-(volume * 5000));
}
```

For at se funktionen i brug, husk at kalde den under **draw()**.

En funktion der bruger en forløkke

Man kan hurtigt få skabt masser af tegning ved at bruge en forløkke; et stykke kode der bliver kørt et bestemt antal gange. I følgende eksempel bruger jeg også funktionen **random()** for at ændre både størrelsen og farven på de ovaler jeg tegner. Derudover gør jeg cirklerne halvgennemsigtige ved at bruge **setAlpha(100)**:

```
function konfetti(){
    let x;
    let y;
    let farve;
    noStroke();
    //tegn 1000 cirkler tilfældige steder
    for (let i = 0; i <=1000; i+= 1){
        x = random(0,width);
        y = random(0,height);
        farve =color(random(0,255),random(1,100),0);
        farve.setAlpha(100)
        fill(farve);
        ellipse(x, y, random(3,16), random(3,16));
    }
}
```

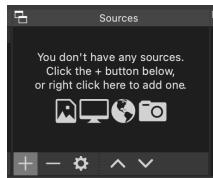
For at se funktionen i brug, husk at kalde den under **draw()**.

Opgave

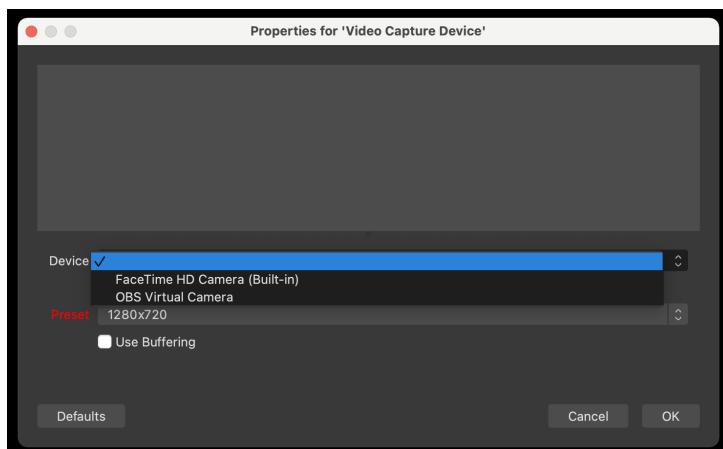
- Prøv at ændre på nogle af tallene og kør koden for at se hvad der sker.
- Leg med udtryk og fyld dit videobillede godt op med tegninger og billeder.

Optimer koden på mac.

I princippet kunne vi godt bare bruge det video-output vi har lavet i p5js, men det fungerer lidt for langsomt til at give en god fornemmelse under et zoommøde, så vi vil hellere optage og bruge video fra OBS. For at gøre det skal vi have tilføjet computerens webcam. Tryk på + for at tilføje en ny **source/kilde** og vælg **Video Capture Device/videooptagelsesenhed**:



Nu dukker en dialogboks op, hvor du klikker "OK". I den næste dialogboks der dukker op skal du under device vælge det kamera der er dit frontkamera og klikke ok.



Nu skulle du gerne kunne se dig selv i OBS.

Særlig til Windowsbrugere

Windows blokerer for at man kan bruge webkamera flere steder på én gang. Se appendix med titlen **Work-around til Windows** for en løsning.

Klargør koden i p5.js - mac optimering

Det var godt at få tegnet videoen sammen med billederne, så vi kunne placere billederne, men nu skal du ikke længere tegne din video under **draw()**. Du får programmet til at springe en linje over, ved at skrive "://" foran. Vi vil gerne have en "greenscreen" som baggrund til tegningerne - det gør det nemt at lægge dem ovenpå i OBS. Så; ændre baggrundsfarven til grøn.

```
function setup() {
  createCanvas(640, 480);
  colorGreen = color(0,255,0);

  firework = loadImage('Firework_transparent.png');

  video = createCapture(VIDEO);
  video.size(240,100);
  //video.hide();
}

function draw() {
  background(colorGreen);
  image(firework, 0, 0, width,300);
}
```



Hvis du gerne stadig vil kunne se dig selv, så kan du under **setup()** sætte størrelsen på video ned, og springe linjen med **video.hide()** over.

kombiner p5.js, OBS og Zoom

I p5.js:

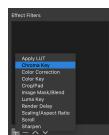
Du skal starte med at gemme din sketch. Under **file** vælges **save**. Vælg et passende navn. Dernæst under **file** vælges **share** og **fullscreen**. Åben linket i et nyt vindue.

I OBS:

Nu skal du tilføje en ny **source/kilde** i OBS. Denne gang skal det være af typen **Window capture/vindue optagelse**. Vælg det vindue der kører din kode

Beskær billedet i OBS ved at holde "option" inde på mac eller "alt" på windows og træk i de røde punkter i rammen. Klip til så det kun er det grønne billede der ses. Du skalerer ved at trække i de røde punkter uden option. Få videobilledet og p5.js billede til at passe så nogenlunde sammen.

Højre-klik på din "window capture"-source og vælg **filters**. Tilføj et nyt **effect filter** ved at klikke på plus og vælg **croma key/croma nøgle**. Default-instillingerne er til greenscreen, så du klikker bare ok.



Nu mangler du kun at starte virtuelt kamera i OBS (under **controls/styring**)

I Zoom:

Dit virtuelle kamera skulle gerne dukke op som et kamera der kan vælges i Zoom. Muligvis skal Zoom genstartes.

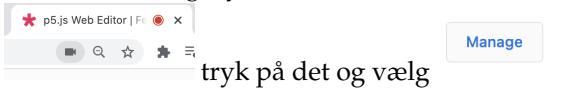
Work-around til Windows

En feature ved Windows, er at man kun kan bruge web-kamera i en app eller browser ad gangen. Det er lidt en udfordring, da vi gerne skulle bruge det uafhængigt af hinanden i hhv p5.js og OBS.

Det første du skal være opmærksom på at at du ikke kan have teachablemachine kørende i baggrunden samtidig med at du bruger OBS. Det næste er problemet med at bruge kamera i p5.js samtidig med OBS.

Løsningen her og nu bliver at starte det virtuelle kamera i OBS, og så bruge det som input i p5.js.

Når du har sketchen åben i Crome og trykker  dukker et lille videokamera-ikon op til



højre i adressefeltet. tryk på det og vælg .

Nu skulle du gerne komme til det sted i settings hvor du kan vælge mellem dit indbyggede



kamera og det virtuelle. Ulempen her er at når der popper et billede op, som følge af dine bevægelser, så genkender den ikke længere bevægelse og billedet bliver taget ned igen. Det giver stadig en sjov effekt, men også lidt benspænd til de kunstneriske udfoldelser.

```

function setup() {
  createCanvas(640, 480);
  colorGreen = color(0,255,0);

  firework = loadImage('Firework_transparent.png');

  video = createCapture(VIDEO);
  video.size(240,100);
  video.hide();

  imageModelURL = 'https://teachablemachine.withgoogle.com/models/0JK9FHm_1/';
}
  
```

Brug af biblioteker

Åbn menuen med dine sketchfiles.

I sketchfiles ligger en fil der hedder **index.html**. Den skal du klikke på. Nu sletter du alt hvad der står, og skriver følgende: (du må gerne copy-paste)

```

<html>

<head>
<meta charset="UTF-8">
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/addons/p5.dom.min.js"></script>
<script src="https://unpkg.com/ml5@latest/dist/ml5.min.js" type="text/javascript"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.2.0/addons/p5.sound.min.js"></script>
</head>

<body>
<script src="sketch.js"></script>
<style>
  body {
    background: #FFF;
  }
</style>
</body>
  
```

Nu vender du tilbage til din kode ved at klikke på **sketch.js**

Det vi har gjort her er, at vi har gjort nogle biblioteker tilgængelige, der ikke ligger som standard i p5.js. Vi skal bruge biblioteket til at klassificere vores video.

Tilføj linjerne med 'label' og 'classifier' til **setup()**:

```

function setup() {
  createCanvas(640, 480);
  colorGreen = color(0,255,0);

  firework = loadImage('Firework_transparent.png');

  video = createCapture(VIDEO);
  video.size(640,380);
  video.hide();

  imageModelURL = 'https://teachablemachine.withgoogle.com/models/qqbeU_Qtv/';
  label = "";
  classifier = ml5.imageClassifier(imageModelURL + "model.json");
}
  
```

classifier er nu bundet op på din specifikke model. **label** er den variabel hvor vi gemmer det som vores videobillede bliver genkendt til.

Vi er der næsten nu

For at få det hele til at virke skal du bruge disse to funktioner. Det er dem der bruges til at identificere billederne fra videokamerareet og gemme resultatet i 'label'. Du skriver dem ind, helt i bunden af din sketch, præcis som de står her

classifyVideo()

```
function classifyVideo() {
  let flippedVideo = ml5.flipImage(video);
  classifier.classify(flippedVideo, gotResult);
}
```

gotResult(error, results)

```
function gotResult(error, results) {
  if (error) {
    console.error(error);
    return;
  }
  label = results[0].label;
  classifyVideo();
}
```

Start classifier

Nu mangler classificeren bare at blive startet under `setup()`. Skriv `classifyVideo()` nederst i `setup`, før den afsluttende `}`.

`setup()` skulle gerne se cirka sådan her ud nu, på nuværende tidpunkt:

```
function setup() {
  createCanvas(640, 480);
  colorGreen = color(0,255,0);

  firework = loadImage('Firework_transparent.png');

  video = createCapture(VIDEO);
  video.size(240,100);
  video.hide();

  imageModelURL = 'https://teachablemachine.withgoogle.com/models/0JK9FHm_1/';
  label = "";
  classifier = ml5.imageClassifier(imageModelURL + "model.json");
  classifyVideo();
}
```